ADST-II-CDRL-STOWA-9900102A
April 30, 1999

ADVANCED DISTRIBUTED SIMULATION TECHNOLOGY II (ADST II)

Synthetic Theater of War – Architecture (STOW-A) 1.6 S/W Baseline Upgrade DO #0032

CDRL AC12

For

Lesson Learned Report

DID: DI-MISC-80711

For:

United States Army
Simulation, Training, and Instrumentation Command
12350 Research Parkway
Orlando, Florida 32826-3224
Attn: AMSTI-EV

By:

Science Applications International Corporation
12479 Research parkway
Orlando, FL 32826-3248

SAIC
An Employee-Owned Company

Lockheed Martin
Information Systems Company
12506 Lake Underhill Road
Orlando, FL 32825

LOCKHEED MARTIN

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** 30 APR 1999	**3. REPORT TYPE AND DATES COVERED** final

4. TITLE AND SUBTITLE Advanced Distributed Simulation Technology II (ADST-II) Synthetic Theater of War Architecture (STOW-A) 1.6 s/w Baseline Upgrade DO #0032 Lesson Learned Report	**5. FUNDING NUMBERS** N61339-96-D-0002
6. AUTHOR(S)	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Lockheed Martin Information Systems ADST-II P.O. Box 780217 Orlando Fl 32878-0217	**8. PERFORMING ORGANIZATION REPORT NUMBER** ADST-II-CDRL-STOWA-9900102A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) NAWCTSD/STRICOM 12350 Research Parkway Orlando, FL 32328-3224	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for Public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(Maximum 200 Words)***

This document presents the lessons that have been learned during the development and operation of the Synthetic Theater of War – Architecture (STOW-A) System. This product evolved over a period of more than four years, spanning both the Advanced Distributed Simulation Technology (ADST) I and ADST II contracts.
In 1994, the Synthetic Theater of War – Europe (STOW-E) demonstration provided the first step towards integrating virtual, constructive, and live exercise components. The system that resulted from the rapid modification of existing simulations was not designed for the set of problems inherent with exchanging data between dissimilar systems that used different protocols and computed different representations of units and basic loads, different terrain, etc. While the basic concept was proven, the system was not stable and had limited functionality.
The first major upgrade to the initial STOW-A system was accomplished to support the Prairie Warrior 95 exercise. The limited virtual simulation and the interface of STOW-E were replaced, and the resulting system proved to be more reliable and extensible, but again the quick turnaround defined by the exercise date limited the ability to design or test the software. Significant improvements were achieved in the correlation of behavior, terrain, and fidelity between the constructive and virtual worlds. User comments and developers' design concepts were collected, correlated, prioritized, and implemented in time for STOWEX 96. Over the next two years, there was sufficient time between exercises to discuss user needs, add functionality, and strengthen stability within the context of improving software design and implementing a controlled test approach.

14. SUBJECT TERMS STRICOM, ADST-II, STOW-A, simulation,	**15. NUMBER OF PAGES** 34
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED**	**19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED**	**20. LIMITATION OF ABSTRACT**

DOCUMENT CONTROL INFORMATION

Revision	**Revision History**	**Date**
-	Initial Submission	03/29/99
A	First Revision	04/30/99

Table of Contents

1 Introduction

This document presents the lessons that have been learned during the development and operation of the Synthetic Theater of War – Architecture (STOW-A) System. This product evolved over a period of more than four years, spanning both the Advanced Distributed Simulation Technology (ADST) I and ADST II contracts.

In 1994, the Synthetic Theater of War – Europe (STOW-E) demonstration provided the first step towards integrating virtual, constructive, and live exercise components. The system that resulted from the rapid modification of existing simulations was not designed for the set of problems inherent with exchanging data between dissimilar systems that used different protocols and computed different representations of units and basic loads, different terrain, etc. While the basic concept was proven, the system was not stable and had limited functionality.

The first major upgrade to the initial STOW-A system was accomplished to support the Prairie Warrior 95 exercise. The limited virtual simulation and the interface of STOW-E were replaced, and the resulting system proved to be more reliable and extensible, but again the quick turnaround defined by the exercise date limited the ability to design or test the software. Significant improvements were achieved in the correlation of behavior, terrain, and fidelity between the constructive and virtual worlds. User comments and developers' design concepts were collected, correlated, prioritized, and implemented in time for STOWEX 96. Over the next two years, there was sufficient time between exercises to discuss user needs, add functionality, and strengthen stability within the context of improving software design and implementing a controlled test approach.

The STOW-A System pioneered many technology concepts, including scalability, translators, network interfaces, wide area network filters, computer generated forces (combat tasks and combat support functions), and the constructive-virtual interface. The following document captures many of the significant lessons learned over the years. While some of the issues described in this document may be viewed as specific to the STOW-A simulations, many will be characteristic of the general constructive-virtual interface complexities.

Many of the "Lessons" are imbedded in the discussions. Where possible, lessons have been specifically identified in the text.

2 STOW-A Overview

STOW-A provides the capability to support unit interactive training and mission rehearsals at the Brigade level. The architecture is based on the Distributed Interactive Simulation (DIS) protocol. The virtual components (semi-automated forces (SAF), translators and filters, 2D/3D viewers and sound system, and the manned simulators) are on the DIS network. Information from the constructive simulation is sent to the constructive-virtual interface function which exchanges data packets between the two simulations.

The constructive simulation used in STOW-A is the Brigade/Battalion Battle Simulation (BBS); virtual simulation is provided by the Modular Semi-Automated Forces (ModSAF) software for SIMNET simulators. The interface linkage between the constructive and virtual simulation is provided by the Operational State Interpreter (OPSIN) and ModSAF while Cell Interface/Adapter Units (CIAU) provide the filter between different layers of virtual simulation as well as the translators between the virtual simulation and the manned simulators. The Application Gateway (AG) is used to reduce network traffic between sites. A 2D map display and 3D stealth provide exercise controllers with both overview and detailed views of the synthetic battlefield. A logger records data for replay and analysis in After Action Review.

The key components for linking the constructive BBS with the virtual manned simulator environments are (1) ModSAF which provides an entity-level virtual representation of BBS units for the simulators and (2) OPSIN which controls the disaggregation of BBS units into ModSAF entities. The OPSIN synchronizes the states of disaggregated vehicles in both the constructive and virtual worlds as events occur in each of them.

2.1 BBS

BBS is a constructive wargame developed for brigade and battalion commanders to train their battle staffs in combat and battlefield operations procedures. The focus of BBS is the command staff. In a typical BBS exercise, the brigade command staff communicate with role players who input commands into the simulation, then report the responses of the simulation back up to the command staff through their normal communication channels.

Constructive level simulation does not represent the forces as individual entities. The control is only at the platoon level or above for each of the units. This control limits the simulation in such a way that it is impossible to determine which tank made a shot or was hit. The only thing that can be determined is that a vehicle in a unit shot or was hit.

2.2 ModSAF

The original purpose of ModSAF was to provide an opposing force for the Simulation Network (SIMNET) manned simulators. ModSAF controls and simulates interactive entities that are organized into units and can be commanded at the company, platoon, and individual level. The orders that are given to the vehicles are represented as tasks. The platoon and company level tasks within ModSAF allow the operator to control larger forces by giving a single order. These orders are then sent to each of the vehicles within the unit. The vehicles can react to a situation individually or as a unit.

Unlike constructive simulations, ModSAF simulates entities at a level of resolution sufficient to support visualization and interaction with humans on the virtual battlefield. The behavioral simulation in ModSAF is more complex than that of constructive simulations since the simulation has to perform obstacle avoidance and find cover and overwatch positions.

ModSAF is generally comprised of a front end processor (plan view display) and any number of back end processors that perform the actual computation. The plan view display (PVD) typically provides the graphical user interface (GUI) that allows an operator to create and task entities or units. For STOW-A, the PVD is used to monitor the exercise movement to insure that no inadvertent activity occurs. The back end processors are responsible for simulating the BBS entities that have been disaggregated. These processors share the same Persistent Object (PO) database as the OPSIN. To achieve the large number of entities created in a STOW-A game, up to 10 back end computers have been used. The optimal number of back end processors is a function of processor configuration/speed and architecture. Too many back ends will actually have a diminishing returns effect.

2.3 Operational State Interpreter

The first implementation of the constructive-virtual linkage was the Advanced Interface Unit (AIU). This device was implemented on a Versa Modula Europa (VME) card cage and then connected to an entity level semi-automated forces (SAF) simulation called ODIN SAF. The AIU used a custom protocol to transfer data between ODIN and a process called SIMCON which ran on the BBS processors. SIMCON used a shared memory segment to communicate with BBS. This implementation allowed units to be controlled by BBS and, when necessary, be replicated using ODIN in the virtual world for interaction with manned simulators. This combination was replaced after STOW-E with OPSIN.

The tasks that are given to the ModSAF vehicles in STOW-A are translated from input given to the BBS system. The device that translates the BBS operational states (OPSTATES) into ModSAF tasks is the OPSIN. OPSIN translates the commands from BBS and sends them to ModSAF. It also is capable of providing feedback to BBS from the ModSAF vehicles. This communication between BBS and OPSIN is a custom network packet protocol. The OPSIN also communicates with the ModSAF machines using the PO protocol. PO protocol creates an abstract database by replicating a large amount of data between the ModSAF machines and keeping that data consistent.

OPSIN is also capable of listening for any other simulators on the network that are using the DIS protocol and on the same exercise. OPSIN will send information about the new unit to BBS based on the information in the DIS Protocol Data Units (PDUs).

2.4 STOW-A Infrastructure

The STOW-A infrastructure is composed of the Cell Adapter Unit (CAU) protocol translator, the Cell Interface Unit (CIU) PDU filter, and the data logger. After Prairie Warrior, these two functions were combined into a single software function and renamed the CIAU, eliminating the need for one computer at each of the remote SIMNET sites.

The CIU was built to filter different kinds of DIS PDUs and was used in several different places along the network. The goal was to lower the amount of bandwidth required on both the local and wide area networks. This device was also used to isolate certain simulations from portions of the network when that simulation conflicted with traffic from another device. The CAU was built to translate between DIS PDUs and the older SIMNET protocol. This translator was a bi-directional device that allowed simulations using different protocols to interact seamlessly.

The data logger was used to capture and play back DIS and SIMNET PDUs and was capable of recording an entire exercise.

3 STOW-A Training Exercises

STOW-A has been used in six major training exercise:

- STOW-E
- Prairie Warrior 95
- STOWEX 96
- WARSTEED 97
- WARSTEED 98

These exercises served two purposes: (1) train the target audience and (2) validate and mature the STOW-A technology. The following section provides a brief overview of the STOW-E, Prairie Warrior, STOWEX, and WARSTEED exercises as the context for the hardware and software experiences. In each case, issues identified during an exercise were either fixed for the exercised, prioritized for future incorporation into the baseline, or allocated to procedural workarounds. The last two STOW-A software releases, version 1.6 and 1.6.1, are described in the next section.

3.1 STOW-E

STOW-E was an Advanced Concept Technology Demonstration (ACTD) funded by the Defense Advanced Research Projects Agency (DARPA). STOW-E combined several types of training in an effort to create a seamless view of the battle. The experiment consisted of live troops on an instrumented range, virtual DIS/SIMNET simulators and computer generated forces, and constructive simulations that do not simulate at the entity level. The original experiment faced several technological challenges, including terrain resolution and correlation between models, differences in the definition of capabilities of the same weapon in different simulations, and the initial task of combining systems that were not built to work together. In addition, the complexity of the system and the number of sites involved created a large logistical and communication problem throughout the development of the project and the actual experiment.

STOW-E also linked to live units in the field. This linkage was accomplished with the Brigade Operations Display and AAR System (BODAS) simulation. This system was connected to the live instrumented range at the Combat Maneuver Training Center (CMTC) in Hohenfels, Germany. BODAS read in data from the range and generated DIS PDUs to be transmitted back to the simulation network.

Another significantly factor with ODIN SAF was the rapidly filling Hash Tables (requiring a re-boot as frequent intervals and the inability of ODIN SAF to assign the same entity ID to an entity following a re-boot. The result was the inability of exercise and tech controllers to monitor and control entities that were killed prior to the re-boot but were alive following the re-boot. The Bumber-Number level of resolution was a much needed capability during STOW-E but could not be fully achieved because of this problem.

The combination of these systems created a single view of the battlefield. This view had limitations, particularly with the live portion of the exercise. However, this experiment proved that the combination of live forces, virtual simulations, and constructive simulations was feasible.

3.2 Prairie Warrior 95

The Prairie Warrior Exercise was the first use of the STOW-A system in the United States. The exercise was also one of the largest uses of STOW-A. Prairie Warrior is the name of the graduation exercise for the Command and General Staff College (CGSC) at Ft. Leavenworth, KS, in May of each year.

This exercise was a Corps level exercise that involved several sites around the world. STOW-A was held during the week prior to the main exercise and was used as a precursor to the Corps Exercise. The goal was to have the forces finish the STOW-A portion of the battle and be in the starting position for the main exercise.

The STOW-A exercise consisted of two Army Brigades (Bde). The first Brigade consisted of three Battalions. The first Battalion was SIMNET simulators from the training facility in Grafenwoehr, Germany. The second Battalion was comprised of units in manned simulators at two SIMNET facilities. The first and second Companies were located at Ft. Knox while the third Company was at Ft. Benning. The Battalion Tactical Operations Center (TOC) was located at Ft. Knox. The third Battalion consisted of four ModSAF workstations located at Ft. Leavenworth. The second Brigade was represented in BBS at Ft. Leavenworth. The command staffs for both Brigades were located at Ft. Leavenworth along with a division response cell. The SIMNET facility at Ft. Rucker, AL, participated with an Aviation Brigade. The Brigade had six AH-4 simulators along with several ModSAF stations to round out the Brigade to full strength. The entire opposing force was represented in BBS at Ft. Leavenworth. This allowed the Blue forces to fight a common enemy on the same terrain.

While the software used in STOW-E had been developed over several years, only a few months were available to procure hardware, replace the virtual simulation, port software to a new platform, install the system in the five locations, design and install the telecommunication networks, design and install the Wide Area Network, and test. Although the schedule was extremely challenging, all hardware and software objectives for the exercise were met. It required a large, highly skilled staff to be resident during the exercise.

The following technical objectives were addressed and achieved:

- Correlation of behavior between entity-based simulation and aggregate-based simulation;
- Correlation of synthetic terrain between constructive and virtual worlds;
- Initial correlation of fidelity between constructive and virtual simulations;
- Passage of Battle Damage Assessment information between constructive and virtual worlds;
- Wide area networking of five field sites.

3.2.1 Hardware Improvements

Changes made to the architecture for Prairie Warrior were focused to make future software development a simpler process. The original VME machines required a special suite of hardware for development, and the ODIN SAF, used for STOW-E, was no longer supported. ODIN SAF did not support DIS, indirect fire weapons, distribution of simulation across multiple simulation hosts, resupply, or the complex terrain databases being used. ModSAF became the virtual simulation for Prairie Warrior and all subsequent exercises. The AIU from STOW-E was ported to UNIX on a Silicon Graphics (SGI) machine and was renamed OPSIN.

In the early development of Prairie Warrior, OPSIN was hosted on an SGI Indy. Since OPSIN interfaces with both BBS and ModSAF, the host computer required two Ethernet ports. In the Indy setup, it did not matter which port was used for communication with the BBS network. However, when the Indy was replaced with an Indigo2, the second Ethernet port would not communicate with BBS. The problem was later determined to be with the new port. The SGI Indy configuration contained an SGI card which added the second Ethernet port. SGI delivered the Indigo2 with a third party card which did not support raw Ethernet mode, the mode required to communicate with BBS. An Intel card and driver were eventually identified and used for the Indigo2 R4400 processors running IRIX 5.3. This problem would resurface when the R4400 and IRIX 5.3 were replaced by the R10000 (R10k) and IRIX 6.2, and again during the porting of OPSIN to an Intel based personal computer (PC).

The problem with Ethernet ports also applied to the CAU when used as a DIS to SIMNET translator since the SIMNET protocol also uses raw Ethernet mode.

3.2.2 Software Lessons Implemented for Prairie Warrior.

The STOW-E disaggregation model was very simplistic, and ODIN SAF technology was inadequate to represent unit movement and interaction. For example, each time a BBS unit was disaggregated into a vehicle representation in ODIN SAF, the supplies on each vehicle were set to full load determined by ODIN SAF at creation. In addition, ODIN created all the vehicles without any damage, although the unit may have previously suffered damage. This immediately prevented any attempt to include supplies in an After Action Review or representing proper unit strengths in aggregation/disaggregation cycles. The notion of basic loads for vehicles, such as M1 battle tanks, was different in BBS and ODIN SAF, and not reconciled. ODIN SAF vehicle movement behavior was inadequate to the task (vehicles often became stuck). The matching of basic loads between the two simulations was a major problem that was improved from release to release.

Because of these and other deficiencies, ODIN was replaced with ModSAF which had better behaviors, and the AIU was replaced by OPSIN. OPSIN established an Ethernet connection with the SIMCON process in BBS in order to receive aggregation/disaggregation commands, and to coordinate the two levels of simulation. A dedicated set of computers running ModSAF was established as part of the hardware architecture. This set of ModSAF machines was called the ModSAF Farm. The Farm communicated with OPSIN through a common distributed database, the PO database. The communication medium between the Farm and OPSIN was based on the DIS protocol rather than on the SIMNET protocol, another reason for replacing the AIU which only used SIMNET.

SIMCON was modified to address the cascading problem encountered in STOW-E where, under certain conditions in the simulation, all the BBS units moved into the virtual world and overwhelmed the virtual simulation capability.

3.2.3 Prairie Warrior Lessons Learned

Analysis of data collected during Prairie Warrior 95 identified five areas requiring improvement: data consistency between the virtual and constructive representation of the battle, reliability of command execution, missing functionality, scalability, and ease of exercise setup and management. These items were addressed after Prairie Warrior in support of the following exercise, STOWEX 96, and continued to be improved in Software Release Versions 1.6 and 1.6.1.

1. Consistency of Virtual and Constructive Ground Truth. The first task of the linkage is to ensure that the virtual representation of the battlefield is consistent with the constructive representation of the battlefield. This is accomplished by sharing data on events and object states within the two simulations. In DIS, this is a fairly easy process since the objects in one simulation are the same as the objects in the other simulation. However, in STOW-A the constructive object is an abstract version of multiple virtual objects. It is harder to track the state of an abstract object via the state of individual objects. The following virtual unit orientation problem noticed during Prairie Warrior indicated inconsistency between the two representations: BBS did not track the facing of units since units are assumed to face the tactically correct direction. When units were disaggregated, they were created facing north. When a unit was put into a static OPSTATE, it was left facing in its last direction. In both cases, the unit's orientation was often tactically incorrect, and it was left vulnerable to attack. ModSAF needed better situational awareness to be used when the unit was first disaggregated or when put into a static OPSTATE.

2. Reliable and Consistent Operation

   - Aggregation/Disaggregation. The processes of aggregation and disaggregation involves the transfer of simulation of an entity between the constructive (BBS) and the virtual (ModSAF) worlds. Criteria of geographic, visual, and electronic proximity, and intention to interact set the parameters for when an entity will aggregate or disaggregate. This criteria is also called a Sphere of Influence (SOI) which is established to trigger the aggregation/disaggregation of constructive simulation controlled entities when they come within proximity of virtual controlled entities to interact. In the Prairie Warrior 95 linkage, the aggregation/ disaggregation procedures were conducted by SIMCON using operator manipulated SOIs which only used geographic proximity for their aggregation/disaggregation criteria. SOIs are either fixed or dynamic, depending upon where their center of mass is attached. During Prairie Warrior, dynamic SOIs did not appear to operate correctly, forcing the use of stationary SOIs. Stationary SOIs require the "game control" operators to monitor all entities to ensure that they do not encounter other entities without disaggregating.

   - Cascading. Problems with aggregation/disaggregation also arose when SOIs allowed a chain reaction of disaggregation to occur. When a unit enters another's SOI, it will disaggregate, creating a new SOI. Units in this second SOI not previously disaggregated will then disaggregate, develop SOIs, and cause disaggregation of units in their SOI. Thus, a disaggregation chain erupts, causing additional entities to disaggregate. Cascading occurred dramatically when a Rotary Wing Aircraft (RWA) or Fixed Wing Aircraft (FWA) flew over multiple aggregated units.

- A unit's combat capability in the virtual world was not degraded when it engaged a unit in the constructive world.

- Aggregation/disaggregation also involves the transfer of all pertinent tactical data when control is transferred. A unit often disaggregated without fuel or ammunition. If the entity was a platoon, it could disaggregate with one of its units identified as an incorrect type (e.g., enemy among friend).

  Disaggregation criteria are important because disaggregated units consume more computational and network resources than aggregated units. The number of disaggregated units is a significant factor that affects the size of the exercise. The Prairie Warrior linkage was limited to very simple disaggregation criteria based solely on geographic proximity. All units were forced to use the same criteria because it was impractical to inform the BBS of the capabilities of individual entities. As a result, too many units were disaggregated. Ideally, the DIS entity should decide when a linkage unit needs to be disaggregated since it has the complete knowledge of its own capabilities and intentions.

  For manned simulators, the predominant disaggregation criteria is what the crew and its sensors should be able to see. All linkage units viewable by the simulator crew must be disaggregated. SAF entities can reason about aggregated linkage units without disaggregating them. Thus, a disaggregation request can be delayed until the SAF entity decides to directly interact with the linkage unit (i.e., shooting at it, exchanging supplies, etc.).

3. Functionality. There were several functional areas not well mapped between ModSAF and BBS, such as the Engineer tasks for minefield bridging and resupply tasks. Some of the loss in functionality was due to the fact that DIS does not support the ability to edit terrain to reflect a temporary change such as a fighting position, berm or ditch. As functionality within ModSAF is increased, the linkage must also be capable of transmitting corresponding data between systems.

   ModSAF did not modify the vulnerability of the vehicles in the disaggregated units. Additionally, BBS has the ability to dismount troops with specific skills, equipment, and supplies. ModSAF models dismounted infantry but not dismounted vehicle crews.

4. Scalability and Robustness.

   - Network Traffic. A single Multiple Launcher Rocket System (MLRS) round consists of multiple (96) warheads, each of which was translated to a detonation packet. Whenever several MLRS rounds were fired in a short period of time, the detonation packets overloaded processing on local machines. Since the bandwidth on the WAN is small, nothing but the detonation packets was transmitted for several seconds.

   - Simulation capability. STOW-A could not generate enough entities to simulate a battle between division level combatants because ModSAF back end machines became overloaded and crashed. The capacity of simulation machines needed to be increased and the software design needed to be improved in order to meet future exercise requirements. There are two aspects to this problem: simulating only the entities needed in the virtual world, and efficiently balancing the virtual entities over the ModSAF machines.

5. Exercise Management and Set Up. BBS cannot process information at the vehicle level; the information must be presented to BBS in terms of the unit to which the vehicle belongs. To perform this mapping, two pieces of information must be given to the linkage: BBS unit IDs and the vehicles that belong to each unit.

Unit organization should be made available on the DIS network by having all units send DIS Aggregate PDUs. Aggregate PDUs would represent linkage units while they are aggregated, including such information as unit marking and center of mass. If all virtual units sent Aggregate PDUs, an application could construct a unit's organization solely from information available on the DIS network.

3.3 STOWEX 96

The purpose of STOWEX 96 was to conduct a brigade level BBS driven Command Post Exercise at Ft. Riley, KS, and Ft. Rucker. Five centers of activity were connected via the DSI network. The NSC at Ft. Leavenworth served as the center for Technical Control. Manned simulators at Ft. Knox and Ft. Rucker participated in the exercise. In addition, a viewport at the Pentagon was provided. Ft. Knox and Ft. Rucker were connected to Ft. Riley by video teleconferencing to support the issuance of orders and tactical coordination between sites. Ft. Riley served as the center for Exercise Control with BBS HICON and workstations that were linked to BBS processors at the NSC via commercial telephone lines. The 1st Brigade, 1ID (Mechanized) at Ft. Riley was the focus of the STOWEX training exercise.

3.3.1 Hardware Improvements and Lessons Learned

One general lesson learned during the software preparation was that each increase in hardware speed introduced software timing issues. This was evident with the ModSAF PVD on a new SGI Solid Impact R10k machine. The machine would crash when the operator issued multiple updates to the task frame execution matrix under system load, regardless of whether the machine functioned as a single front end controlling multiple ModSAF back end machines or as a pocket system. Although the code had not changed from previous versions, the problem was manifested only on R10ks under load and with quick succession of operator tasks. The PVD displays the tasks as an execution matrix. The operator was able to make changes to the future tasks and tell the units to continue to the next (future) phase. The system crashed when the operator made changes, told the unit to continue to the next phase, and then immediately pushed the "done" button. The problem was that "done" was a higher function than the changes made to the matrix. If the system was busy, it was possible for the "done" to be executed before the task changes were made to the PO database. The system would process the done button, then try to execute tanks that did not exist.

3.3.2 Software Lessons Implemented for STOWEX 96

The transition between Prairie Warrior and STOWEX 96 provided major changes in functionality and architecture. It also began to establish BBS as the keeper of the authoritative databases for supplies and unit/vehicle status. BBS continued to maintaine operational control of its units. The functions that SIMCON performed were moved into BBS code, allowing BBS and OPSIN to communicate directly with each other.

OPSIN was redesigned to have an internal core consisting of ModSAF libraries and a shell specifically designed to interact with BBS. With this design, constructive simulations other than BBS can use OPSIN and ModSAF, modifying only the shell. Raw Ethernet packets are used as the communication media between BBS and OPSIN, and the communication between OPSIN and ModSAF and the rest of the world is done on a DIS network. The specific packets used to communicate with BBS or other constructive simulation are translated into internal OPSIN packets, again to correspond with the core/shell construction of OPSIN.

A protocol was created between OPSIN and BBS to define packets of data for unit creation, aggregation and disaggregation, unit status, and other packets to coordinate the different levels of simulation. The major feature of these packets is that they communicate unit level damage status and unit level supply status. For example, the creation packet BBS sent to OPSIN includes the number and type of vehicles in the unit as well as the quantity of each munition by vehicle type. OPSIN receives this packet, builds the vehicles in the PO database, and distributes the supplies dictated in the create packet. Algorithms were built into OPSIN to make the vehicle supply distribution as uniform as possible.

A major difficulty to overcome was that the naming system for types of vehicles and supplies in BBS was different from the naming system in ModSAF. BBS uses ASCII strings to identify types while ModSAF uses DIS enumerations. Since the ModSAF naming system uses a recognized standard, it was decided to use DIS enumerations in the communications between BBS and OPSIN. This forced BBS to build the correspondence with DIS.

At the start of the game, BBS sends OPSIN a create packet for each BBS unit. OPSIN creates each of those units in the PO database that it shares with the ModSAF Farm, but the vehicles are not simulated by the Farm until BBS sends OPSIN a disaggregation sync packet. Between creation and disaggregation or between two subsequent disaggregations of a unit, the unit is simulated in BBS and may suffer losses and expend some of its supplies. The intent of the disaggregation sync packet is to adjust for these differences. It includes the status of vehicles by type, i.e., two M1 mobility killed and two M1 dead, and the remaining munitions by type of vehicle. OPSIN then processes this packet to set the proper vehicle status and to redistribute the supplies by vehicle type, again uniformly distributed, and then ModSAF begins to simulate the vehicles in the unit. At this point, BBS has relinquished simulation control of the unit to ModSAF, but still maintains operation control over the unit by sending tasking packets to OPSIN which translates those packets into ModSAF tasks and enters them into the PO database for the unit.

Since BBS releases simulation control over a disaggregated unit, any changes in state of the units when simulated by ModSAF must be reported to retain currency of BBS databases. OPSIN assumes the task of keeping BBS concurrent by using the PO database to accumulate the supplies and status of the vehicles in a disaggregated unit and report the summary to BBS as a unit status. BBS takes appropriate action to adjust its database.

The STOWEX software added the Aggregate State PDU (ASPDU) which broadcasts the vehicle membership of a unit and its hierarchy structure over the DIS and SIMNET networks. This had a major effect on the method used to inform BBS of the vehicles and units not created by BBS such as any manned simulators and all of the "Blue" ModSAF forces. In previous versions, these entities were planned by exercise control, and a data file was created to be read at the start of the game. This data file recognized vehicles only, and not any hierarchical structure. With STOWEX, the Blue ModSAF forces could transmit their ASPDUs. OPSIN uses the incoming ASPDUs to identify these units, using the Entity State PDUs (ESPDUs) to identify and group vehicles into their respective units, and reporting the units, number, and type of vehicles in the unit to BBS. When subsequent ESPDUs come in for the vehicles in one of the non-BBS units, the status of the vehicle is cached by OPSIN. OPSIN periodically rolls up all the cached vehicle data into a status report for the unit and sends it to BBS.

Another capability introduced by ASPDUs was related to the visualization of aggregated units on display devices. Before this version, there was no way to display an aggregated unit because the vehicles in the unit were not being simulated, which means that no ESPDUs were sent for the vehicles. With this version, OPSIN assumed the task to send out ASPDUs for BBS aggregated units, allowing their visualization, even on the stealth.

These new packets which included vehicle status and supply information provided a start to an AAR package driven by the BBS databases. However, one important element missing in this version was the tracking of supplies of non-BBS units. That information is not typically transmitted in the standard DIS protocol. STOW-A release 1.6 subsequently introduced an experimental DIS packet to handle this problem.

Another set of functionality introduced for STOWEX 96 was increased minefield capability. A minefield creation packet was added between BBS and OPSIN to allow the creation and deletion of BBS minefields in ModSAF. Upon receipt of one of these packets, OPSIN would adjust the PO database to either create a new minefield or delete an existing minefield, and the ModSAF back end machines would adjust the new data introduced into the PO database.

3.4 WARSTEED

WARSTEED was the 2nd (US) Infantry Division Brigade Combat Team (BCT) external evaluation training program utilizing a combat training center model that included a trained/dedicated opposing force (OPFOR), an observer controller and evaluator (OCE) package as well as STOW-A simulation architecture.

The concept of operations from the WARSTEED Exercise Directive states that the WARSTEED Exercise was to consist of two phases. The first phase involved the 4-7 Cavalry and the second phase involved the 1st Brigade Combat Team. The second phase marked the first time that the Division and the Korea Battle Simulation (KBSC) and the Warrior Training Center (WTC) had conducted an integrated live operation, constructive simulation (BBS), and virtual simulation (SIMNET) linking Task Force operations in the field to a Task Force fighting in BBS as well as a Task Force fighting in SIMNET. This combination of live (via "swivel chair" operation), constructive, and virtual simulations was provided by STOW-A. The second phase of the exercise consisted of two rotations. During the first rotation, TF 2-72 AR fought in BBS, TF 1-72 AR fought in SIMNET, and TF 2-9 IN fought live in the field. During the second rotation of the second phase of the exercise, TF 2–9 fought in BBS and SIMNET while TF 1-72 fought live in the field. The second rotation did not include STOW-A.

3.4.1 Lessons Learned During Exercise

Observations about exercise planning, operational efficiency, degree of technical control, and maximum use of system capabilities were made during support of the WARSTEED 97 exercise.

Observation 1: Refreshing the system on a regular basis is an important part of normal operations. Care must be taken to schedule these refresh periods at times when tactical conditions permit (i.e., when there is no engagement or disaggregation). Refresh operations were scheduled and executed in a timely and correct manner for the most part; however, certain components of the system were not refreshed when they should have been.

Recommendation: The Technical Control Leader should call for the refresh of ALL components of the system, not just OPSIN and the ModSAF Farm during the scheduled refresh period. This includes Blue ModSAF pockets, STRIPES, and CIAUs. Several times during the exercise, a prior refresh would have prevented a component from failing. It is recommended that the NSC review the Tech Control procedures/checklist by NSC to ensure that this procedure is covered.

Observation 2: Recording exercise logs is a very important task allocated to the personnel in Tech Control. These logs are useful for AAR as well as for diagnosing system problems that occurred during an exercise. On several occasions, one log was allowed to span more than a 24 hour period. It is also critical during a lengthy exercise to maintain the 9GB disk and forecast when it is going to fill up with log data. When the drive reaches 75% capacity, it is time to start pulling log files off onto tape. While care was taken to assign people to almost every other workstation in Tech Control, the Data Logger did not have anyone assigned to it.

Recommendation: WTC should ensure that there is one person dedicated on a part time basis to ensure that logs are recorded and that free space on the 9GB drive is maintained. It is sufficient to check the status of the machine once every 20 to 30 minutes to ensure that data is being logged correctly. Disk space can be checked once per 24 hour period, always maintaining a 25% free space buffer. Blank tapes should be made available prior to the start of the exercise for exercise log archiving purposes. Also, a new log file should be started every night at midnight (during 24 hour operations). It is useful to include the date in the log file name. It is recommended that the NSC review the Tech Control procedures/checklist by NSC to ensure that this procedure is covered.

Observation 3: There are two technicians at the WTC who are responsible for maintaining the system (separate from the SIMNET technicians). The WTC technicians are continuing to become more familiar with the system and specifically the IRIX operating system. Users of the system in Tech Control are able to draw on the expertise of these two technicians when a problem or question arises. However, the AAR analysts do not have a technician in the room with them because the technicians (one per shift) must focus their attention on technical control issues.

Recommendation: Since the AAR analyst's main focus is on identifying the data that needs to be collected from a tactical standpoint and not on the intricacies of operating the AAR system, a dedicated technician with expertise in STRIPES, IRIX, EXCEL, and DOS should be available on at least one shift to help with the more technical aspects of the data collection process. Additional SGI and PC training may also be in order for the AAR analysts. WTC should either provide additional training for the AAR analysts or provide personnel that already have this knowledge to aid the AAR analysts.

Observation 4: During the system preparation phase, several tasks were accomplished by the STOW-A support staff during the eleventh hour that could have been performed at a more appropriate time. These tasks included installing three revised ModSAF reader files, installing two revisions of the terrain database (one after the other), and adding two Blue ModSAF pockets to the SIMNET network. In each of these cases, it was apparent that these labor intensive items could have been done more effectively by performing these tasks earlier in the system preparation phase. The later the change, the higher the risk associated with the change. This risk is a result of reduced testing time available at this point in the schedule. When all of the factors were analyzed, the root of the problem appeared to be the late development of the exercise plan.

Recommendation: The exercise plan needs to be developed as early as possible. Ideally, the exercise plan should be available early enough so that the BBS database can be generated in time to be reviewed at the OSF prior to arriving on site to support the exercise. This item is key to the success of an exercise. Late development of this item forces everything else (e.g. BBS database, terrain scrubbing, hardware requirements, etc.) to be accomplished too close to the beginning of the exercise and therefore introduces risk. The implementation and use of an Exercise Control Plan such as developed for Prairie Warrior 95 would improve this situation.

3.4.2 Software Baseline

The STOW-A software baseline for the Korea Hub Site at Camp Casey was largely based on the software baseline used during STOWEX 96. Changes to the STOWEX 96 were made to accommodate both defect corrections and Chorwon Terrain. Table 3..2-1 summarizes these versions.

Table 3.4.2-1 WARSTEED Software Baseline

Tape Name	STOW-A Function	S/W Version
CIAU Version 4.2.5 Field Installation	xciau2, xciau4	4.2.5
OPSIN/ModSAF Version 1.97.1 Field Installation	OPSIN	1.97
ModSAF v2.1.1 STOW-A Version	Farm PVD, all ModSAF Farm, all SIMNET ModSAF	2.1.1 (STOW-A)
STRIPES-4.2.3 Field Install	STRIPES 2D, 3D, and Analysis (in both AAR Room and Tech Control), data logger in Tech Control	4.2.3
Chorwon Terrain Database (Format 5)	farmpvd (install in /usr/people/modsaf/common/terrain), all ModSAF Farm (same location as farmpvd), all SIMNET ModSAF Pockets (same location as farmpvd), OPSIN (same location as farmpvd), STRIPES 2D (/usr/people/stow/stripes-4.2.3/its/terrain)	05-10
Max Impact IRIX 6.2 Patches	STRIPES 3D in Tech Control and AAR Room (for installation instructions see 'INST' tool in IRIX on-line documents)	IRIX v6.2 patches
Phobos fe Driver v2.10	OPSIN	2.1

4 STOW-A Development Lessons Learned

The following section describes the software development of the 1.6 release and the 1.6.1 update. This effort differed from the events described in the STOW-A Exercise section above. Although the goal of Version 1.6 was to support training at Ft. Stewart, the schedule supported the software development timeline as well as training.

4.1 STOW-A v1.6

The modifications made to the STOWEX software baseline provided a significant upgrade in both functionality and stability. ModSAF was upgraded from 2.1 to 3.0, and the improved resupply capability advertised for ModSAF 3.0 was intended to provide improved logistical consistency between BBS, ModSAF, and the manned M1 and M2 simulators. In addition, minefield, RWA/FWA, and limited icon consistency issues identified during STOWEX 96 were addressed. The improved STOW-A software was released as version 1.6.

The 1.6 release also provided significant improvements in the performance of the system. ModSAF memory leaks were identified and fixed, allowing the system to stay up significantly longer. Previously, exercises had to be designed around the known limitations of the system. While an exercise load of 500 vehicles had been a long standing goal of the user, the schedules for STOW-E, Prairie Warrior, and STOWEX 96 did not provide the time for the extensive data analysis and testing cycle required for stability improvement. Six weeks were added to the development schedule for stability improvement, with the result that the system stayed up under laboratory conditions for over 6 hours. Machines could even be left running ModSAF over night without crashing.

The major changes for Software Release Version 1.6 included:

- Incorporation of ModSAF 3.0, including new supply model,
- Creation of the Supply Status PDU,
- Development of the Legacy Application Adapter,
- Consistent fuel and munition tracking,
- Tailgate resupply,
- Corrected simulator mapping files for SIMNET-Training sites,
- Improved stability.

4.1.1 Memory Leaks

A memory leak is a deficiency in a program that prevents it from freeing up memory that it no longer needs. As a result, the program grabs more and more memory until it finally crashes because there is no more memory left. The issue of ModSAF memory leaks was a known contributor to the STOW-A stability problem. As indicated previously, STOW-A was unable to support a large exercise for more than 2 or 2 ½ hours. One of the most obvious problems was the PVD which could not support multiple zooming actions.

ModSAF is a large and complex program, developed by different organizations and many programmers over a span of several years. Like STOW-A software, ModSAF development frequently had to meet schedule deadlines. After a delivery, the priorities were on fixing critical problems and addressing new requirements. While programmers frequently want to go back and correct design deficiencies, there is seldom the time or budget to support this. The attention given to stability by the NSC and STRICOM for the 1.6 release was an exception to this unfortunate norm that allowed significant memory leak problems to be identified and corrected.

The software tool Purify was used to identify actual and suspected memory leaks in the STOW-A components that used ModSAF libraries. This included not only STOW-A ModSAF and OPSIN but also STRIPES. The major deficiencies identified by Purify were in libpvd (the PVD zoom problem) and in PO database handling of taskframes. In the case of libpvd, zooming in or out created a list of aggregates each time a zoom ocurred. The list generate by previous zoom action was not reused and was never released back to the OS. With PO database taskframes, some behaviors, in particular libmove, did not clean up some subtasks, and the problem would worsen with multiple aggregations and disaggregations. Purify idientified locations in memory that were never deallocated as well as multiple pointers to the same locations in memory.

Identification and correction of memory leaks was an extensive and iterative period of running Purify, finding leaks, testing corrections, and rerunning Purify. Since some of the original usage of pointers to the PO database was done for performance, changes needed to be tested very carefully to ensure that there were no ripple effects. The obvious lesson is that there are definite benefits resulting from spending the time and resources to clean-up the design on a semi-regular basis.

4.1.2 Logistical Consistency

One of the two major functional upgrades required for 1.6 was Logistical Consistency. Originally, this concept was intended to track the changes in logistical data (fuel and munitions) that occur when supplies are decremented due to fuel and munition expenditure during an exercise and when those supplies are replenished during a resupply mission.

4.1.2.1 General Resupply Issues

Implementing the resupply function uncovered many problems that required successive iterations of test-fix-test to fully uncover. Version 1.6 addressed tailgate resupply using only the M966, M978, and M979 resupply vehicles. Corrections for service station resupply tasks were deferred to 1.6.1.

For a STOW-A exercise, the resupply mission must be capable of working from BBS, across OPSIN, within DIS ModSAF (Farm), across the CIAU and the AG, with SIMNET ModSAF (Site Blue ModSAF), and with an LAA. This required a rigorous approach during development to assure that each facet was isolated and tested to identify specific changes required for each component.

The initial attempts to perform resupply missions identified the following:

- Neither tailgate nor service station resupply missions worked correctly;
- ModSAF crashed when an M977 munitions carrier was given a tailgate resupply task;
- ModSAF crashed when an M977 was given a tailgate resupply task with the OSF simulator on the network;
- Some resupply functions could be performed in pocket mode but resupply with multiple ModSAF machines configured as front end/back ends (sometimes referred to as "network mode") crashed ModSAF;
- The munitions and fuel accounting did not work correctly and the supply vehicles did not decrement their stores;
- Not all vehicles within a platoon would be resupplied by an M977 loaded with an adequate number and correct type of munitions;
- Loading a ModSAF M977 via pallets adversely affected ModSAF performance (since each pallets was an individual entity), and could cause a crash;
- Smoke rounds could not be zeroed out (in order to be resupplied).

Agreements made with the NSC and STRICOM eliminated the pallet problem. It was agreed that supply vehicles would be loaded in BBS prior to disaggregation. However, this implementation could not be checked out until the resupply OPSTATES were mapped to the ModSAF 3.0 taskframes.

More detailed issues resulting from ongoing resupply testing included:

- The inability to save resupply scenarios;
- Not all vehicles in a platoon were resupplied;
- The ammo supply vehicle could only be loaded with 150 of anything (any combination of ammunition from large missiles to small bullets. This necessitated a request to the NSC to provide munition volumes. The arbitrary 150 limit was removed from ModSAF and reader files were updated.
- Accountability (decrement supply vehicle stores correctly) continued to be inconsistent. OPSIN round-off errors were corrected, but a discrepancy of several gallons of fuel on the M978 remained. One cause was ultimately traced to the fact that the M978 incorrectly consumed fuel from its storage tank.
- Basic ModSAF wheeled vehicle maneuverability problems caused the supply vehicle to crash into the vehicle requesting supplies (either a ModSAF vehicle or a simulator).
- The supply vehicle occasionally stopped short of its resupply point.

4.1.2.2 SAF Design Changes

The transition from version 1.5 to 1.6 provided major changes in the modeling of supplies in ModSAF and increased ModSAF functionality for supply mission. To accommodate these change, OPSIN had to modify its interface to the ModSAF supply model. Changes in ModSAF minefields and minefield breaching caused OPSIN to introduce new packet definitions between BBS and OPSIN.

The new supply model in ModSAF 3.0 required extensive testing and corrections, but eventually provided accurate loading and tracking of supplies on vehicles, including the ability to define the components of the vehicle and track the status of those components. Components such as electrical systems, engine systems, radio components, etc., can be defined for specific vehicle types by reader files. For STOW-A, this definition was accomplished for only a few critical vehicle types such as M1 tank types, and M2 vehicle types. More importantly, this more realistic model defined the allowable capacities for fuel and munitions on all vehicle types. Before this change, ModSAF 2.1 allowed OPSIN to load any munition in any quantity onto any vehicle. To use the new ModSAF model effectively, the definitions of basic loads for vehicle types in BBS and ModSAF had to match more closely, otherwise munitions would be lost and not tracked.

The new supply model provided a fuel tracking capability because the new model tracked fuel consumption and defined fuel capacities for vehicle types. These capacities provided data to distribute a unit level fuel quantity down to the individual ModSAF vehicles in the unit. Changes were made to the unit creation, disaggregation, and unit status packets, and supporting code in both BBS and OPSIN for fuel tracking.

Previous to this version, manned simulators were always considered as individual vehicles, not part of any unit. New functionality added by STOW-A to the basic ModSAF 3.0 allowed manned simulators to be included in non-BBS units and to account for their munition and fuel changes. The module which introduced this functionality is the Legacy Application Adapter (LAA). This functionality allows BBS to interact almost directly with manned simulators in CSS operations as described below.

The STOW-A capability to perform resupply missions allows virtual supply trucks, operationally controlled by BBS, to resupply all units in the simulation, even units that include manned simulators, providing a seamless connection between the constructive simulation, BBS, and the manned simulators. Each end of the connection "sees" the resupply from its own perspective. The manned simulator actually sees the supply truck pull up to the simulated tank and provide supplies. The constructive simulation sees the supply status of the supply vehicle being decremented. In order to maintain a consistent authoritative supply database, only BBS can initiate resupply missions.

The BBS system has the capability to task supply vehicles to do a tailgate resupply mission as well as a service station resupply mission. For a tailgate resupply mission, BBS disaggregates a single supply vehicle in the area of the resupply, then sends a supply packet indicating a tailgate resupply mission to the vehicle. As a result, the single supply vehicle checks for vehicles within a default radius that need the supplies that the supply vehicle has to offer. The supplier vehicle then travels to each requester and dispenses the supplies.

For a service station resupply mission, BBS disaggregates a single supply vehicle or a unit of supply vehicles and sends a supply packet to the vehicle(s) indicating a service station resupply mission. As a result, the vehicles assemble in the indicated position and sit waiting to be approached by vehicles needing to be resupplied

Once the supply vehicle has been disaggregated, BBS controls the rest of the operations by exchanging packets with OPSIN that initiate the resupply action. OPSIN translates those packets into supply loading and tasking data by communicating with the ModSAF Farm using the PO database. The supply packet causes the supply vehicles to be loaded with the supplies determined by BBS. The second packet is a unit movement packet which causes the supply unit to be tasked to do the tailgate resupply mission or service station resupply.

The units being supplied are either BBS units or non-BBS units. If they are BBS units, BBS will send OPSIN a tasking packet appropriate for resupply. Non-BBS units are tasked by agents other than BBS to perform the same tasks.

The new supply model also provides another new capability. Previously, supply tracking could be done only on those units created by BBS because those vehicles were in the Farm PO database and their supply data was available to OPSIN through the Farm PO database. Non-BBS units are simulated by machines that do not use the Farm PO database, so another mechanism was designed to obtain supply data. This mechanism, a new experimental DIS PDU, was introduced in ModSAF and called the Supply Status PDU (SSPDU). The SSPDU broadcasts the supply changes on a vehicle whenever a change takes place. OPSIN receives the new PDUs and, knowing which vehicles are in which non-BBS unit, rolls up the supply changes into a unit status report which informs BBS about the aggregate vehicle damage as before and the aggregate supply status, both by vehicle type. The implementation of code was event driven to minimize increased PDU traffic. The SSPDU is sent out (1) every 15 minutes if no change in supply status is perceived, or (2) upon munition or fuel change. Although a change of a single round or 10 gallons of fuel is sufficient to trigger an update, there is a designed delay of 10 seconds to allow for successive rounds of rapid fire to be included in a single SSPDU rather than send out multiple SSPDUs that have insignificant changes. Note that any of the threshold values may be changed in the reader files.

Creating the new SSPDU instead of incorporating the data in the ASPDU was the better design choice. Including supply data in the ASPDU would have overloaded (incorrectly expanded) the stated purpose of the ASPDU which is to track aggregates. Updates for supply data to the aggregate state PDU would have caused updates to the ASPDU not related to aggregates. Further, confusion would have been introduced with a task such as “follow the simulator” when the LAA would have created an aggregate composed of the simulator and SAF units simulated on different machines.

4.1.2.3 SIMNET Resupply Issues

Once the resupply function was corrected sufficiently to be able to resupply a DIS entity from another DIS entity, the next step involved testing SIMNET-to-SIMNET resupply and DIS-to-SIMNET resupply. Although DIS-to-DIS and SIMNET-to-SIMNET resupply worked, DIS-to-SIMNET resupply failed. Since DIS-to-SIMNET involves the translation of PDUs between the two protocols, the initial culprit was assumed to be the CIAU. When the CIAU was verified to be working correctly, it was found that the declarations in the DIS PDU capabilities field were exactly reversed from the corresponding one in the SIMNET PDU.

4.1.2.4 Manned Simulator Resupply Issues

In addition to the problems encountered within the basic ModSAF 3.0 supply model, there were several issues resulting from resupplying manned simulators. Multiple trips to the Ft. Knox Mounted Warfare Simulation and Training Center (MWSTC) were required to achieve accurate resupply of the M1 and the M2 simulators. Limited resupply testing could be accomplished at the OSF with the LSE M1 simulator and the SIMNET M1 located with CCTT, but there were no M2 simulators in Orlando.

The first attempt to resupply the OSF SIMNET M1 from ModSAF running in SIMNET mode resulted in crashing ModSAF. Over a series of several attempts, it was verified that ModSAF would crash as soon as the supply vehicle was given the resupply task. The cause of the crashing was corrected, but ModSAF was unable to transfer supplies to the simulator. Without a functional MCC or an M1 that was physically capable of supporting the manual steps necessary for resupply in the OSF, the integration effort shifted to the MWSTC to (1) understand the existing resupply capability of both the M1 and M2 simulators and (2) log the data created during an MCC resupply in order to determine the structures of the simulator supply request and the supply vehicle offer.

The initial integration effort at the MWSTC identified munitions differences between the M2 simulator and ModSAF, DIS to SIMNET translation problems, M2 simulator resupply constraints, and decrementing problems for the M977 (ammunition HEMMT). Corrections to these problems included resurrecting the little known art of how to modify the SIMNET mapping files, decoding how the old MCC performed resupply for the M2 simulator (different from the M1), and learning the manual procedures involved in resupplying an M2 in order to derive timing data for the resulting modification required in ModSAF.

LESSON: Do not assume that simulators with the same protocol (e.g., SIMNET M1 and SIMNET M2) perform the same operation in the same fashion.

LESSON: Allow double the anticipated time when testing with manned simulators. No two operate the same. Allow for down time due to crashing, recalibration, configuration differences (software files), and unscheduled reprioritization of the asset.

The resupply goals for the second trip to Ft. Knox included:

1. Verifying resupply of the M1 and M2 by SIMNET ModSAF,
2. Verifying resupply of the M1 and M2 by DIS ModSAF (through a CIAU),
3. Verifying resupply of multiple simulators leading ModSAF M1 and M2s (adding an LAA to the configuration).

The resupply of the M1 simulator worked as expected, i.e., the correct number of supplies were received by the M1 but there accountability problems with the ModSAF supply vehicle persisted. The testing of the M2 uncovered some unexpected features. Resupply of the M2 simulator is significantly different from a ModSAF M2. Where the ModSAF M2 requests rounds of 25mm munitions (HE and/or AP), the SIMNET M2 requests 1 canister and receives 30 rounds iteratively over a span of several minutes. A ModSAF-to-ModSAF resupply will load a large number of rounds instantaneously. Not only was the "1 = 30" subtlety an elusive concept to grasp, the difference in resupply timelines posed a difficult problem. There is significant soldier-in-the-loop intervention because the M2 simulator requires manual procedures similar to those used in the actual BFV to load munitions (turret rotation, switch settings, munition type selection). Not only would ModSAF load 1 round in response to the request for 1 canister, but it would timeout if the manual procedures were not performed fast enough. The "design" needed to be expanded to determine if the M2 was a SIMNET simulator or a ModSAF entity in order to know how many rounds to give in response to a request, and the ModSAF resupply timing had to be adjusted.

When the test of resupplying the simulators by the DIS ModSAF was performed, the M2 simulator issued the service request, the M977 HEMMT issued the resupply offer but for the wrong munition, and no supplies were transferred. The M1 issued the service request, but the M977 did not offer supplies.

The corrections to ModSAF to address the M2 resupply problems included:

- Adding a workaround for an M2 simulator so that ModSAF would offer 30 (rounds) in response to a request for 1 (canister);
- Increasing the timelines associated with loading M2 HE and AP rounds as well as TOW missiles;
- Added correction to ModSAF performed by Burlington that corrected the DIS-SIMNET capabilities problem identified in Orlando;

LESSON: It cannot be assumed that the implementation of a function in the SAF follows the same approach as used for the same function in the simulator; not only are there operational differences between vehicle types (i.e., M1 and M2 request and accept supplies differently), but also the design approach to resupplying an M2 SAF vehicle may differ from resupplying an M2 simulator, and the design approach of resupplying an M1 simulator may differ from that of an M2 simulator.

There is a significant amount of time required to resupply manned simulators. The SIMNET vehicles take 40 seconds to resupply each round of M1 munition and each round of 25mm M2 munition, and 60 seconds for each TOW missile. Loading 26 rounds in the M1 took approximately 30 minutes. In addition, ModSAF resupply capability does not support efficient use of multiple M977s (a second M977 will go to all of the same vehicles that the first M977 went to). Consequently, the combination of the long timeline to resupply manned simulators and the inability to use multiple ammunition supply vehicles means that it will take several hours to supply several simulators via tailgate.

LESSON: Implementing service station was deferred from 1.6 due to design issues. However, the resulting timeline to perform tailgate resupply for manned simulators is prohibitive during most training exercises. The operational requirements and timelines of both approaches should be considered as well as the design implications.

4.2 STOW-A v1.6.1

The update to Version 1.6 (Version 1.6.1), included (1) miscellaneous fixes to design and functionality, (2) implementation of service station resupply deferred from Version 1.6, (3) implementation of DIS 2.04, and (4) rehosting of the software from SGI machines to an Intel based platform (PC Rehost). During a technical interchange among developers, customer, and users, design improvements and functionality corrections were discussed, assessed, and prioritized for implementation. These included:

- Service Station: Incorporate the Service Station function from ModSAF 4.0. The goal was to use a LogPac and allow the operator to determine what supplies to load. The supplies would then be distributed evenly among cargo HEMMTs. (As implemented, BBS disaggregates a single supply vehicle or a unit of supply vehicles and sends a supply packet to the vehicle(s) indicating a service station resupply mission. As a result, the vehicles assemble in the indicated position and sit waiting to be approached by vehicles needing to be resupplied.)
- K-kill issue (deficiency)
- Unit lay down issue (deficiency: difficulty in killing a vehicle that has already been mobility/fire power killed)
- Clone issue (deficiency)
- Artillery visuals/munitions damage mapping
- PO reduction: ModSAF is spending too much time managing the PO created by OPSIN to represent the BBS game rather than simulating vehicles. Modify OPSIN to delay the creation of the objects until necessary, thereby increasing Farm processing capability and system scalability.
- Farm Network Switch: All Farm machines share a single 10Mb connection. Replacing the hub with an 18 port switch will increase effective bandwidth. Of the three performance constraints previously identified (memory, network, cycles), the memory problem has been addressed (1.6 stability). The switch addresses the network issue.
- SIMNET ModSAF MICLIC (deficiency).

- UENEMY Task Optimization: OPSIN's approach to developing sighting reports for BBS was an inefficient search and match procedure. A new library had been written to replace this with a callback methodology.

- Logger time synched with game time (Ft. Knox issue and AAR issue).

- BBS OPSIN protocol logger (debugging tool for problems between BBS and OPSIN).

Of these items, the most significant lessons learned derive from the PO Reduction task which was a major software architecture redesign and the PC Rehost task. The efficiency gained from PO reduction, the increased processor speed of the PCs, and the increased bandwidth provided by the 100Mb switch combine to give STOW-A the long sought after scalability. Once the PC rehost task is hardened, STOW-A will be capable of supporting a much larger exercise than previously possible.

4.2.1 PO Reduction

PO database reduction is a significant change introduced into 1.6.1. Previous to this version, all the BBS units were created, but not simulated in the ModSAF Farm. The entire unit structure was placed in the PO database, including the vehicles in the unit, and was disaggregated only by request from BBS which controlled all aggregation/disaggregation decisions. Using the functionality built into the PO database, disaggregation/aggregation is basically setting or unsetting a bit the data structure of a vehicle. This implies that some units in the Farm PO database may never be disaggregated, but the PO database still bore the expense of carrying them.

In this version, the PO database overhead was reduced by the following changes. At startup, only the PO entity that represented the top level of a BBS unit was entered in the ModSAF Farm PO database; it remains in the PO database throughout the game. A request to disaggregate the unit then causes the creation of the rest of the PO entities needed to round out the unit; a request to aggregate the unit causes the deletion of all but the top level of the unit. The disaggregation sync packet sent from BBS to OPSIN causes the creation of the vehicles and lower level hierarchy entities with states of the damage and supplies and unit formation read from the BBS database.

4.2.2 PC Rehost

One of the major scalability constraints for STOW-A has been the processor speed of the ModSAF and OPSIN platform. SGIs have been used since Prairie Warrior 95, starting with Indys and progressing through processor improvements of the Indigo2 R4400 to the R10k. The price of a STOW-A suite has always been expensive since the R10k price ranges around $24k with 320MB of memory and a 4GB hard drive. The hardware investment of a Farm and OPSIN alone is nearly $200k. The goal of the PC rehost task was to port ModSAF and OPSIN to Intel based PCs to reduce cost and increase performance.

The ADST II ModSAF Delivery Order had previously ported ModSAF to PCs running Linux. However, STOW-A has specific issues that are not usually encountered during typical ModSAF usage. Constructive-virtual linkage uses more ModSAF back ends than other applications, and OPSIN requires a second Ethernet port. As described previously, this second Ethernet connection to BBS has caused difficulties in the past. In addition, the design decision was made to use the Solaris operating system (OS) rather than Linux or Windows NT because of its stability. The Government wanted to provide the flexibility for the user and wants an OS that is not vendor restricted. Solaris has product support, and a Sun OS was viewed as an advantage because many sites were getting Sun workstations. The PCs and the Linux OS used for STOW97 required the development of several drivers. Obsolescence was also a concern: the version of software ported to a specific PC configuration today might have to be updated for a different configuration bought in the future.

To minimize the problems inherent with loading the Solaris OS on PCs, the PC vendor was required to deliver PCs with Solaris configured and loaded on the machines. In addition, all PCs were configured with a second Ethernet port. This provided maximum flexibility and supported the optional task of porting the CIAU or AG to a PC. However, the problems encountered with the communication between BBS and OPSIN on this second port precluded the option to rehost the CIAU or AG.

The PCs were ordered in two separate procurements. The first set of 5 were ordered with dual 400 MHz processors with 256 MB of memory and 2 Ethernet ports. The second set of 5 were ordered with the same memory but were dual 450 MHz processors.

The initial rehost of ModSAF to the PCs encountered some software porting problems due to byte swapping differences between SGIs (big endian) and PCs (little endian). Although the baseline ModSAF had previously been rehosted to PCs, STOW-A encountered rehost issues for several reasons. One significant problem was that of byte alignment. The SGI compiler aligns words on 64 bit boundaries which is required for the MIPS processor in the SGI. The Solaris compiler aligns words on 32 bit boundaries by default for the Intel processor in the PC. This is not a problem for the PC port itself, until data is read that has been structured and created by a machine using the other alignment. This is what caused much of the problems in the ModSAF port to the PC. The C7L terrain files were created by the c7b_to_c7l tool on an SGI machine that used 64 bit word boundaries. This caused problems when sol86_ModSAF was loading the terrain and also when attempting to access terrain data for runtime use. In both cases the misalignment caused memory access to occur at the wrong address:offset which would return what appeared to be corrupt data. In actuality, the data was fine, but it was the address:offset that was incorrect. The problem of alignment along with byte ordering also caused problems with the PO. This problem did not occur for the Linux port of ModSAF because the GCC compiler was used. The GCC compiler has a compiler option to generate code using 64 bit alignment. The Sun Solaris x86 compiler does not have this option, therefore each structure had to be manually padded so that words were aligned on 64 bit boundaries. In addition, a bug was found in the c7b_to_c7l code which caused a corrupt c7l terrain file to be created. Once this problem was found, it was fixed in the STOW-A baseline. Further coordination with the ModSAF Delivery Order verifiedthe problem had already been fixed in the ModSAF baseline. Note that the STOW-A 1.6.1 release was based on ModSAF 3.0. At the time of the PC Rehost task, ModSAF was releasing Version 5.0, although the byte ordering and byte alignment issues still exist in ModSAF v5.

The second difference between STOW-A ModSAF and the baseline ModSAF lies with the fact that, unlike STOW-A, most ModSAF users do not use the PO database. Thus several new byte swapping and byte alignment issues were encountered. One terrain issue was traced to a byte swap problem in the compilation. The PCs and SGIs use different file extensions on the terrain database files(C7L vs. C7B to designate the byte ordering, L for little-endian, B for big-endian Buildings were improperly defined in the data structures, causing problems on the PVD.

The most complex rehost issue involved the second Ethernet port. The Solaris network socket is implemented differently than the one on the SGI. The OPSIN/BBS linkage requires raw ethernet communication. In addition to the typical TCP and UDP protocols, SGI provides a raw ethernet protocol. Because of this, the connection to OPSIN can be obtained/maintained via the socket/ioctl interfaces. Solaris does not provide this raw ethernet protocol, so DLPI capability (which allows raw ethernet access) had to be added to OPSIN to use on the BBS side. DLPI network code from ModSAF was reused as a starting point, then modifications were performed to work with BBS. The buffering issue was solved by adding a configurable streams module called "bufmod" that buffers network traffic. The socket dropped packets from the hardware buffer before OPSIN could look at them because the buffer was not large enough to hold data that OPSIN had not yet processed. The solution was to modify the manner in which the OS handled the buffer, effectively increasing the size of the buffer. This solution is a platform-specific fix. This problem was particularly difficult to characterize since OPSIN was able to receive some but not all of the BBS data.

Individually, STOW-A ModSAF and OPSIN were successfully ported. Only limited testing was performed on the PC Farm with the PC version of OPSIN because the effort was performed in parallel with the testing required for the Version 1.6.1 release and resources were limited. There were approximately two crashes that occurred that were analyzed but the problem was not definitively identified. It is suspected that the crashes may be timing issues resulting from using the faster processor, similar to the problem in version 1.6 described earlier when the R10k was substituted for the R4400 on Blue ModSAF. Specific problem identification and the resulting solution will require additional testing to pinpoint the cause. The 1.6 problem (PO Find Entry) was observed rarely over the several month development period, but occurred repeatedly during the extensive Alpha testing. This repetition was required in order to identify the specific conditions present when the crash occurred.

The PCs were shipped to Ft. Stewart at the end of 1998 to demonstrate the improved performance to the NSC and STRICOM. While the PCs were successfully used as Blue ModSAF machines (and thus displayed the faster performance and potential for larger loads), the attempt to use them as OPSIN and a Farm was unsuccessful. Although the link lights indicated that BBS packets were being received, OPSIN could not see the BBS traffic. Without the opportunity to trouble shoot this further in Orlando, the problem was assumed to be related to site specific network hardware.

One other problem encountered with the PCs at Ft. Stewart was the inability to communicate with the SIMNET simulators. This could not be tested further, and is assumed to be a byte swap issue. It is very possible that this rehost issue was not previously identified by the baseline ModSAF effort since ModSAF does not use SIMNET mode extensively.

4.3 Software Design Decisions and Implications

The SAF used for STOW-A has been an evolutionary project. To produce the different versions, many design decisions were made and later reconsidered as new technology or new requirements were introduced, or as the old design was proven to be insufficient.

4.3.1 Representation Compatibility

By their very nature, constructive simulations and virtual simulations have very different representations of the units and objects that they simulate. If a multi-resolution simulation is developed from two or more existing simulations, as the STOW-A SAF has been, the differing representation problem must be addressed. In addition to that basic difference is the specifics of the representations in BBS and ModSAF which must be resolved. For example, a 120 mm Sabot round is the same type of munition in both systems, but BBS refers to that type by an ASCII string, while ModSAF uses a standard DIS enumeration.

It should be pointed out that representational problems do come in at least two broad categories, anticipated and discovered. While all representational problems should be anticipated over the length of thorough requirement analysis and design development, the *de facto* STOW-A approach discovered them only by performing integration tests and running exercises.

4.3.1.1 Using DIS Enumerations

As the STOW-A SAF evolved, it became obvious that the supply load assigned to units by BBS and the vehicle loads assigned to the vehicles that represented the disaggregated version of the unit should be closely correlated. This is necessary for meaningful AAR and to preserve fidelity throughout the representations. Since BBS used ASCII strings to identify munition and vehicle types, and ModSAF used standard DIS enumerations, a standard nomenclature was adopted which used DIS enumerations in the packets between OPSIN and BBS. This forced the building of translation tables in BBS to convert DIS enumerations to the BBS internal nomenclature. This was not as easy a task as it appeared, and omissions and transpositions were a constant problem. Contributing to the problem was the fact that ModSAF was itself not completely compliant to the DIS standard. This precluded using a standard document as a reference.

The first set of translation tables was based on the DIS 2.03 standard, and it was assumed that this version of DIS would always be used. However, to support interoperability with a wider set of simulations, the STOW-A version was adapted to use either 2.03 or 2.04. The translation tables had to be modified to handle this option, and the same type of problems resurfaced. No easy tool was developed to create these tables or to verify them.

4.3.1.2 Basic Load Compatibility

As the sophistication of the STOW-A SAF increased, it became apparent that the basic loads of munition and supply should be controlled. BBS has a great deal of flexibility when assigning supplies or defining the armaments on vehicles in their units. In fact, a unit can be charged with fuel which exceeds the capacity of their fuel tanks. ModSAF, on the other hand, is much more constrained. The vehicle models define the fuel and ammunition capacities, and those constraints are not easily ignored. Because ModSAF posed the greatest constraints, the BBS database was modified so that the vehicles that its units contained would be mapped easily into ModSAF vehicle types. Upon disaggregation, OPSIN had the responsibility to distribute the unit supplies to the individual vehicles created to represent the unit in the virtual world. BBS determined the amount of supplies from the state of the unit in its database, and sent OPSIN the amount of supplies classified by vehicle type. OPSIN then distributed the vehicle type supplies uniformly among the vehicles of that type, ignoring those vehicles with a status of DEAD or MOBILITY-FIREPOWER KILLED. Fuel presented a special case because BBS would send fuel capacity by entire unit, not by vehicle type. OPSIN distributed fuel among all vehicles in the unit, using vehicle fuel tank capacity as a weighting factor. This distribution of supplies occurs each time a unit is disaggregated and in effect represents a crossleveling of supplies.

Just as in the case of converting BBS to use DIS enumerations, the matching of basic loads and armaments was an involved process which changed data files for both ModSAF and BBS. Since the vehicle models were controlled by reader files, they can theoretically be modified on site without the aid of the development team. However, changes to data files must be treated as carefully as changes to software. Not only do changes to a released baseline need to be reviewed and approved, but also they must undergo the same level of testing to verify correctness and evaluate their effect.

4.3.1.3 Main Gun Round Representation

In addition to making the basic load of the models in both simulation programs, there were also some peculiarities to be accounted for. During the creation of combat units in BBS, e.g. units containing M1A1 tanks, only one type of round was assigned to the unit. For example, 160 rounds of 120mm TK would be assigned to a 4 M1A1 platoon. In ModSAF, the M1A1 main gun basic load consisted of two types of rounds, M830A1 and M829A2, high explosive and Sabot rounds. It was the responsibility of OPSIN to distribute the 160 main gun rounds to 64 rounds of M830A1 and 96 rounds of M829A2. OPSIN further distributed the rounds by type uniformly to each of the M1A1 vehicles created for the unit. Other instances of this procedure took place for other vehicle types, and the data necessary for this main gun round distribution was captured in a reader file. In this file, the distribution data was indexed by gun ammo type, i.e., munition type A is split in to munition types A, B, C, using the distribution factor a, b, c, where $a+b+c = 1$. When sending a unit status packet to BBS, OPSIN added the number of rounds in each type for the M1A1 vehicle, and reported the sum. This implies that, over subsequent disaggregations, a unit that has fired some of its main gun rounds could change the distribution of main gun rounds without further changes in the remaining total main gun rounds.

4.3.1.4 Probability of Kill

Although most of the representational problems occurred between BBS and ModSAF, some also occurred between ModSAF and manned simulators. Basic loads of the M1 manned simulator and that of ModSAF had to be reconciled. A less obvious problem was related to kill and damage probabilities. Both ModSAF and manned simulators have internal tables that determine the amount of damage a vehicle suffers when hit. The table is indexed by the type of the impacting munition and the location it hits on the vehicle; the results are determined probabilistically. The results should be the same for M1 tanks created and simulated by ModSAF and manned simulators representing M1 tanks.

Often the results were noticeable different. Damage caused by mines in a minefield to M1 tanks was resolved by similar tables, and again the results between ModSAF and manned simulators was different. Tables such as these and the code that evaluated the damage based on these tables had to be resolved in order to maintain simulation fidelity across the different types of simulators, BBS, ModSAF and the manned simulators.

4.3.1.5 Mobility Differences between ModSAF Vehicles and Manned Simulators

As exercises were run, mobility differences between ModSAF vehicles and manned simulators became apparent. ModSAF vehicles could go places where manned vehicles could not. For instance, slopes that exceeded a certain angle could not be climbed by manned simulators, but ModSAF vehicles could easily negotiate them. ModSAF vehicles could not maneuver well in tight spaces, and took too much time to perform tailgate resupply or could not traverse minefield breaches well. The solution was to modify ModSAF to improve wheeled vehicle (HEMMT) maneuverability for resupply. Different design approaches to improve the ability of a vehicle to go through a breach lane with expected success were discussed. A "perfect breach" was the easiest implementation but not operationally accepted, so the alternative approach of widening the breach lane was employed. The differences between climbing hills was not addressed since this was an issue related to the Korean Chorwon terrain database used during Prairie Warrior.

4.3.1.6 Mobility Differences between BBS Units and ModSAF Units

There were mobility differences between BBS and ModSAF at a unit level. An aggregated BBS unit has accessibility to more terrain than the same disaggregated unit has. Often a unit would disaggregate in no-go terrain and not be able to maneuver as a disaggregated unit. Part of this problem is due to the differences in terrain representation between the BBS terrain database and the ModSAF database, part is due to each program's mobility criterion, and yet another part is due to resolution differences between the two terrain databases. It is a very complex problem which was addressed sufficiently to allow the exercise to proceed without a major impact on exercises. For new terrain databases, an analysis to assess compatibility across simulations is recommended. In worst case, terrain may need to be created from a common source for the different applications.

4.3.1.7 Spatial Resolution Differences

BBS and ModSAF use different scales of measurement for positional coordinates and area measurements. ModSAF typically has a one meter resolution, and BBS typically has a ten meter resolution. This did not pose significant problems when disaggregating units at a location specified by BBS, because center of mass of the unit would be placed at the location. The actual vehicles were distributed about the location depending upon the formation and the soil type. For example, code was introduced in OPSIN to adjust the position of individual vehicles to avoid water and no-go terrain, as well as trying to keep all of the vehicles on one side of a body of water. In this case, the center of mass of the disaggregated unit would not be at the location specified by BBS. It was far more important to model a reasonable laydown of vehicles rather than strictly adhere to specified locations.

For the most part, operations that depend upon point locations were in fact specifying an event that takes place in an area around the specified point location. Modeling BBS artillery strikes in the virtual world, modeling BBS minefields, locations of supply service station, and tailgate missions are translated to area operations in ModSAF.

One unusual problem related to rendezvous of a vehicle to a disaggregated unit remained unresolved, but posed no significant problems. BBS would task a vehicle to rendezvous with a unit by tasking the vehicle to move to the location of the unit. When this is translated into ModSAF (because the unit and vehicle were disaggregated), the vehicle may not find any vehicles of the unit near the specified location. This is because the vehicles in the unit were spread out due to the action experienced while disaggregated.

Spatial Resolution problems were more noticed in the modeling of BBS minefields.

4.3.2 Decisions affecting PDU Traffic

A major factor in any large scale simulation using DIS protocol packets is the amount of PDU traffic between the simulators. Each simulator must process the packets directed at it, and can be over loaded when the traffic becomes too large. As the STOW-A SAF passed through its evolutionary stages, attempts were made to reduced the size of the traffic that was observed in previous stages.

4.3.2.1 Firing MLRS Rounds

In Prairie Warrior exercises, BBS executed artillery missions which fired several MLRS rounds from outside a disaggregation zone to inside. Because the rounds were detonated inside of a disaggregation zone, OPSIN translated the artillery mission into corresponding detonation packets in DIS which would be broadcast to the rest of the virtual world. Unfortunately, there is a multiplier effect when using MLRS rounds, because each round disperses a large number of bomblets, and each of the bomblets produces a detonation PDU which is broadcast. Such a multi-round MLRS mission caused a large spike in PDU traffic broadcast to all the ModSAF farm machine, to all the ModSAF non-farm machines, across Gateways and translators, and finally to manned simulators. In hindsight, this problem should have been anticipated, but it was discovered just prior to the exercise. To keep the Prairie Warrior exercise going, code changes were made to map the individual MLRS rounds into a 500 pound bomb to produce roughly the same scale effect in the virtual world.

4.3.2.2 Playboxes

The areas in which virtual battles were fought, i.e., battles between disaggregated BBS units and other virtual units, were generally well defined subareas of the entire battlefield over which BBS operates. BBS, in the constructive world, was able to create entities and objects over the entire battlefield, but many of those object would not affect the areas in which the virtual battles were fought. Minefields is a good example. Minefields which BBS created outside of the virtual battle areas would not affect the virtual battles. However, the virtual representation of a BBS battlefield produces PDU traffic, much of which is unused by the virtual world. In order to prevent the simulation in the virtual world of some of those objects, the concept of playboxes was introduced. A playbox was a collection of rectangles imposed on the battlefield and associated with a class of objects, such as minefields. These playboxes were used by OPSIN to decide to create an object, such as a minefield, and represent it in the virtual world. If the object was inside of the playbox, it would be created. Usually a playbox was defined by a command line argument entered when starting OPSIN; this allowed only one rectangular region per object and did not use the full capabilities of playboxes. The coordinates, the diagonal corners, of the playbox were decided upon during the planning of the exercise. These playboxes were associated with two classes of objects: minefields and dead hulks.

4.3.2.3 Dead Hulks

Dead hulks were created to give some reality to damage on the battlefield after a battle between aggregated BBS units. In such a battle, if a unit suffers a loss of a vehicle, BBS would send OPSIN a packet which would take a vehicle in the (now unsimulated) unit, disassociate it from the unit, set its appearance to a dead hulk, and begin to simulate it. This procedure would produce dead hulks in the virtual world which broadcast entity state PDUs on the network. Manned simulators or non BBS units that entered the battle area would then see the dead hulks, that is, receive the PDUs of the dead hulks. The first attempts of this concept immediately demonstrated that dead hulks were being laid down in the virtual world outside of the virtual battle areas. This, of course, resulted in sending out unnecessary entity state PDUs. Just as for minefields, a playbox was associated with dead hulks to decide to reduce the affect on performance.

4.3.2.4 Optimum Number of Back End Processors

During the Prairie Warrior exercise, network traffic among the ModSAF Farm machines seemed to be a problem. The Farm at that time consisted of 10 machines acting as back ends. The Farm seemed to be unstable and often process bound. It was speculated that part of the problem was due to the fact that a great deal of traffic was related to keeping the shared PO database current among all the machines. Each change in the PO database by one Farm machine had to be broadcast to and processed by all the other Farm machines. To experiment with this hypothesis, the number of machines was reduced to eight, and the Farm behavior became more stable. No definitive experiments were ever carried on to establish the validity of the hypothesis, but the number of Farm machines was never more than eight.

4.3.2.5 Transmission rate of Aggregate State PDU

Since the aggregate state PDU provides information for an entire unit and is usually displayed only when the unit was aggregated, the rate of transmitting the PDU does not need to be as fast as the transmission rate for entity state PDUs. It was decided that the heart beat rate of the aggregate state PDU should be 30 seconds, much slower that the heart beat rate of entity state PDU. A mechanism was built into OPSIN and ModSAF to control both the heart beat rate and time out rates of aggregate state and entity state PDUs. In addition, a dead reckoning routine was coded for the movement of the symbols used for aggregate units.

4.3.2.6 Persistent Object Database Reduction

As originally designed, OPSIN created a complete representation of each BBS unit when the game started. BBS create packets, sent at the start of the game, contained enough information to determine what types of vehicles and how many of each type were in the unit. OPSIN used the information to create all the vehicles for the unit at start up. The vehicles stayed in the PO database, usually for the duration of the exercise. When the unit was aggregated, the vehicles were not simulated. When the unit was disaggregated by BBS, OPSIN would essentially "flip a simulation bit" in the unit, and all the vehicles would then be simulated. While this description is an over simplification, it captures the essence of the process. When the unit was again aggregated by BBS, the simulation bit for the unit was reset, causing the Farm to stop simulating the vehicles. The vehicles remain in the PO database as unsimulated objects.

This technique has many advantages, such as vehicle persistence and constant name (vehicle binding), but it also had a large disadvantage: it produced a very large PO database to be maintained by the ModSAF Farm, and the resulting PO database traffic caused problems as discussed above.

STOW-A software version 1.6.1 changed the technique of using the PO database. First of all, at the start of the game, the creation packets were used by OPSIN to create in the PO database only the root PO object of the unit, not the entire unit. (The PO database representation of a unit contains PO objects for all the vehicles and for the hierarchy structure of the unit.) Only when BBS commands a disaggregation is the rest of the unit created in the PO database and inserted into the hierarchy under the existing root object. Following this creation, the simulation of the vehicles begins by the ModSAF Farm. The disaggregation-sync packet, which triggers this event, has the types of vehicles and numbers of each type as well as the present damage and supply status of the unit as known by BBS. That provides enough information to produce an accurate representation of the unit. When the unit is reaggregated, OPSIN deletes all objects in the PO database associated with the unit except the root object. If disaggregation occurs again, the same creation process is repeated.

Although this technique caused some problems, especially with dead hulks, the savings in PO traffic ultimately allowed more entities to be simulated by the ModSAF Farm.

4.4 Mapping

Consistent and complete mapping of units, vehicles, and supplies through all of the STOW-A components was a never ending problem. Extensive mapping coordination and testing was performed for each exercise and each major software release that required visual checkout on BBS, the ModSAF PVD, STRIPES, the SIMNET PVD, the SIMNET Stealth, and the SIMNET M1 and M2 simulators. The coordination involved the expertise of the user to determine which set of vehicles and munitions were important for a particular exercise, and the testing required the participation of a subject matter expert who could correlate the nomenclature with icon symbology and visual representation. As a minimum, a table should be developed that includes a description of each entity, the associated symbol, and an image of the associated visual model. If a method to update each component were devised, update and testing time could be reduced.

4.5 Hardware Lessons Learned

4.5.1 Network

When designing a STOW-A network for installation at a site, care must be taken to isolate the network from traffic outside the scope of a STOW-A game. It is often necessary to determine from the site technicians whether an external node broadcasting traffic is present on the STOW-A network. The designer must take care to ensure that the BBS ring is not connected to any external nodes other than OPSIN. This has been extremely problematic in past installations and exercises.

The SIMNET subnetwork has been another source of potential performance problems in the past. It is important during network design to ensure that no more than three SIMNET ModSAF machines are connected through a single vampire tap to the SIMNET Thicknet backbone. This can be accomplished by making sure that no more than three ports are used on any MT-800 concentrator (or like equipment).

4.5.2 Cabling

For the 10-Base-T portions of the STOW-A network, a good quality CAT-5 Ethernet unshielded twisted pair (UTP) cable such as Belden's "Datatwist Five" M/N #1583A should be used. These cables should be terminated with CAT-5 RJ-45 connectors. It is also beneficial to check all custom-made cables before use with a good cable tester that will verify continuity as well as CAT-5 level of performance. This is especially true for long cable runs such as between the Battle Simulation Center and SIMNET buildings in the Korea installation.

For long network runs, fiberoptic cable may also be used as it was in the Ft. Stewart installation. When purchasing components to interface with an existing fiberoptic cable installation, it is very important to know exactly what type of cable (FDDI, Single Mode, Multi-Mode, etc.) and terminating connectors (ST, SC, etc.) are present.

4.5.3 IP Addressing Schemes

A standardized approach to selecting IP addresses for the different subnetworks in STOW-A has been adopted. By adhering to the convention used in the following '/etc/hosts' file, consistency across STOW-A sites can be ensured:

```
#  IP address-hostname database (see hosts(4) for more information).
# Default IP address for a new IRIS. It should be changed immediately to
# t
he address appropriate for your network.
# (The '192.0.2' network number is the officially blessed 'test' network.)
# This entry must be present or the system will not work.
127.0.0.1       localhost
# Assigned multicast group addresses listed in RFC-1060 ("Assigned Numbers").
# These entries can be deleted if you don't want them.
# (They are also available via the Internet DNS name servers.)
224.0.0.1       all-systems.mcast.net
224.0.0.2       all-routers.mcast.net
224.0.0.4       dvmrp.mcast.net
224.0.0.5       ospf-all.mcast.net
224.0.0.6       ospf-dsig.mcast.net
224.0.1.1       ntp.mcast.net
224.0.1.2       sgi-dog.mcast.net
224.0.1.3       rwhod.mcast.net
224.0.2.1       rwho.mcast.net
224.0.2.2       sun-rpc.mcast.net
#######################################################################
#
#       /etc/hosts file for OSF Lab
#       Date: 08/21/97
#
#     Network Diagram
#
#     **************          *************************
#     ** BBS(DEC) **          **  DEVELOPMENT SYS    **
#     **************          *************************
#           |                            |
#     *********************   **************************
#     ** (DEC)OPSIN(200) ** - ** BBS-ModSAF FARM(200) **
#     *********************   **************************
#                                     |
#                              ** XCIU2-DIS **
#                                     |
#                           ********************
#    ** XCIU4-DIS **--------*** Main DIS 201 ***
#           |               ********************
#           |                         |
#    *****************          ** AG-DIS **
#    ** SIMNET(XXX) **                |(NULL Modem)
#    **     M1      **        ** AG-Remote **
#    **  Stealth    **                |
#    **  ModSAF     **     ***********************
#    *****************     ** Ft. Overhill(202) **
#                          ***********************
#
#
#######################################################################
#
######### Fort Underhill ###############################
#
#       Application Gateway from DIS Net
#
192.9.201.30    AG-DIS                ag-dis
164.217.1.1     AG-DIS-REMOTE         ag-dis-remote
#
```

```
#       Application Gateway from Remote Net
#
192.9.199.31    AG-REMOTE               ag-remote
164.217.2.1     AG-REMOTE-DIS           ag-remote-dis
#
#       Remote ModSAF at Ft. Overhill
#
192.9.199.10    R-ModSAF1               r-modsaf1
192.9.199.11    R-ModSAF2               r-modsaf2
192.9.199.12    R-ModSAF3               r-modsaf3
192.9.199.13    R-ModSAF4               r-modsaf4
192.9.199.14    R-ModSAF5               r-modsaf5
#
#       XCIU2 PO Filter between BBS and DIS
#
192.9.201.92    XCIU2-DIS               xciu2-dis
192.9.200.92    XCIU2-BBS               xciu2-bbs
#
#       XCIU4 Translator between DIS net and SIMNET
#
192.9.201.94    XCIU4-DIS               xciu4-dis
192.9.222.94    XCIU4-DIS-SIMNET        xciu4-dis-simnet
#
#        Simnet ModSAF Machines
#
192.9.222.10    SIMNET-ModSAF1          simnet-modsaf1
192.9.222.11    SIMNET-ModSAF2          simnet-modsaf2
192.9.222.12    SIMNET-ModSAF3          simnet-modsaf3
192.9.222.13    SIMNET-ModSAF4          simnet-modsaf4
192.9.222.14    SIMNET-ModSAF5          simnet-modsaf5
192.9.222.15    LAA                     laa
192.9.222.4     SIMNET-LOGGER           simnet-logger
#
#       Blue ModSAF on the DIS Network
#
192.9.201.11    ModSAF1                 modsaf1
192.9.201.12    ModSAF2                 modsaf2
192.9.201.13    ModSAF3                 modsaf3
192.9.201.14    ModSAF4                 modsaf4
192.9.201.10    ModSAF5                 modsaf5
#
#       Stealth(3D), PVD(2D), Logger, and Sound
#
192.9.201.2     TECHPVD                 techpvd
192.9.201.3     TECHSTEALTH             techstealth
192.9.201.4     LOGGERTECH              loggertech
192.9.201.5     TECHSOUND               techsound
#
#       OPSIN BBS Network
#
164.217.20.10   OPSIN-BBS               opsin-bbs
#
#       BBS-ModSAF Farm Network
#
192.9.200.10    OPSIN-ModSAF            opsin-modsaf
192.9.200.11    BBS-ModSAF1             bbs-modsaf1
192.9.200.12    BBS-ModSAF2             bbs-modsaf2
192.9.200.13    BBS-ModSAF3             bbs-modsaf3
192.9.200.14    BBS-ModSAF4             bbs-modsaf4
192.9.200.15    BBS-ModSAF5             bbs-modsaf5
192.9.200.16    BBS-ModSAF6             bbs-modsaf6
192.9.200.17    BBS-ModSAF7             bbs-modsaf7
192.9.200.18    BBS-ModSAF8             bbs-modsaf8
192.9.200.19    BBS-ModSAF9             bbs-modsaf9
192.9.200.20    BBS-ModSAF10            bbs-modsaf10
192.9.200.21    FARMPVD                 farmpvd
#########################################################################
# D. Burton - 10/22/96 - Needed for PPP modem connectivity
#########################################################################
```

```
127.0.0.1       localhost             loghost
192.0.2.2       pppserv.domain.com    pppserv
192.0.2.3       pppcli.domain.com     pppcli
```

4.5.4 SGI Machine Administration, Execution Scripts

For convenience, it is best to use scripts to call up the executables on the different STOW-A machines. Through experience, the following scripts have evolved and are in use at most of the STOW-A installations:

start PVD:

```
cd /usr/people/modsaf/common/src/ModSAF
./modsaf_sgi_6_2 -nosim -terrain ntc-0101 -aspdu -exercise 7 -database 10 -dis -articdr
```

start your farms:

```
#!/usr/bin/sh
###############################################################################
#
# START_YOUR_FARMS
#
# Last Modified:
#    07/06/98   D. Burton
#
# Purpose:
#    This script is executed from BBS-ModSAF1.  Its purpose is to start seven
#    ModSAF Farm back ends.  Before doing this, it checks to make sure that
#    there are no xwsh processes running (remote xterms from ModSAF Farm back
#    ends) on BBS-ModSAF1.  It then calls up an xterm for each of the seven back
#    ends and executes an rlogin as user 'modsaf' for BBS-ModSAF2 through
#    BBS-ModSAF7.  The .login file for the user 'modsaf' on BBS-ModSAF2 through
#    BBS-ModSAF7 contains a call to the shell script 'start-back' which causes
#    ModSAF to be executed with all of the correct parameters.  On BBS-ModSAF1,
#    a call is made directly to ModSAF, since it will be running locally.
#
###############################################################################

# Kill all ModSAF processes currently running
./kill_your_farms

cd /usr/people/modsaf/common/src/ModSAF

# Start modsaf on BBS-ModSAF1 (ModSAF running locally)
xwsh -bg black -fg cyan -sl 1000 -geom 40x29-1000-1000 -title BBS-ModSAF1 -e
./modsaf_sgi_6_2 -nogui -terrain ntc-0101 -aspdu -exercise 7 -database 10 -articdr &

# Start modsaf on BBS-ModSAF2 through BBS-ModSAF7 (ModSAF running remotely)
xwsh -bg black -fg cyan -sl 1000 -geom 40x29-450-1000 -title BBS-ModSAF2 -e rlogin bbs-
modsaf2 &
xwsh -bg black -fg cyan -sl 1000 -geom 40x29-1-1000 -title BBS-ModSAF3 -e rlogin bbs-
modsaf3 &
xwsh -bg black -fg cyan -sl 1000 -geom 40x29-1000-1 -title BBS-ModSAF4 -e rlogin bbs-
modsaf4 &
xwsh -bg black -fg cyan -sl 1000 -geom 40x29-450-1 -title BBS-ModSAF5 -e rlogin bbs-
modsaf5 &
xwsh -bg black -fg cyan -sl 1000 -geom 40x29-1-1 -title BBS-ModSAF6 -e rlogin bbs-modsaf6
&
xwsh -bg black -fg cyan -sl 1000 -geom 50x29-400-200 -title BBS-ModSAF7 -e rlogin bbs-
modsaf7 &
```

start-back:

```
#!/usr/bin/sh
###############################################################################
#
# START-BACK
#
# Last Modified:
#    07/06/98    D. Burton
#
# Purpose:
#    This script is executed from each of the ModSAF Farm back ends (BBS-ModSAF2
#    through BBS-ModSAF7).  Its purpose is to execute ModSAF locally with the
#    correct parameters.  Before doing this, it checks to make sure that
#    there are no ModSAF processes currently running from previous executions.
#
###############################################################################

# Kill all ModSAF processes currently running
VAL1=`ps -e | grep modsaf`
kill $VAL1
killall rlogin

cd /usr/people/modsaf/common/src/ModSAF

# ./modsaf_sgi_6_2 -terrain ntc-0101 -exercise 7 -database 10 -articdr -aspdu -nogui
./modsaf_sgi_6_2 -terrain chorwon_022097 -exercise 7 -database 10 -articdr -aspdu -nogui

cd /usr/people/modsaf
```